This listing of claims will replace all prior versions and listings of claims in the application:

LISTING OF CLAIMS:

1. (Currently Amended) A method ~~executed in a computer system~~ for automatically tracking build information comprising:

extracting build information from one or more builds wherein said one or more builds are produced using a compilation process;

registering said one or more builds by storing said build information corresponding to each of said one or more builds;

automatically determining ~~at~~ runtime software module information about a version of software being tested; and

automatically determining a first of said one or more builds corresponding to said runtime software module information.

2. (Original) The method of Claim 1, further including:

storing build information in a database; and

using said build information in said database to automatically determine said first build corresponding to said runtime software module information.

3. (Original)The method of Claim 2, wherein said database that includes said build information is an object database, and the method further comprises:

creating and storing one or more objects corresponding to each of said builds; and

creating and storing one or more objects corresponding to software modules included in each of the builds.

4. (Original)The method of Claim 3, further including:

creating and storing a session object corresponding to a test session of said version of software;

creating and storing one or more objects corresponding to software modules describing said runtime software module information;

automatically determining a previously created build object corresponding to one of said one or more builds previously registered; and

storing an address of said previously created build object in said session object.

5. (Original)The method of Claim 1, wherein said registering one or more builds includes:

storing software module information about at least one module for each of said one or more builds.

6. (Original)The method of Claim 1, wherein said automatically determining at runtime software module information about a version of software being tested further includes:

gathering runtime information about software modules dynamically loaded in said computer system.

7. (Original)The method of Claim 6, wherein subroutine calls associated with an operating system executing in said computer system are used in said gathering runtime information.

8. (Original)The method of Claim 2, further comprising:

using said build information in said database to automatically determine a second of said one or more builds corresponding to software module information included in a bug report; and

creating and storing a bug report object corresponding to said bug report, said bug report object including an address associated with said second build.

9. (Original)The method of Claim 8, further including:

submitting said bug report included in a formatted electronic message;

interpreting data included in said formatted electronic message to enable said determining of said second build.

10. (Original)The method of Claim 9, further including:

creating and storing another build object corresponding to a build in which said bug report is identified as being corrected; and

associating said other build object with said bug report object in said database.

11. (Original)The method of Claim 1, wherein said automatically determining said first build further comprises:

determining said first build for which a matching build is being determined from said one or more builds registered;

determining a candidate list including one or more builds having at least one module in common with said first build;

for each build included in said candidate list, determining if modules included in said each build match modules included in said first build;

for each build included in said candidate list, if there are no modules included in said each build that have a module name and associated attributes matching a module included in said first build, determining that said each build is not a match for said first build; and

for each build included in said candidate list, if a first of said modules included in said each build has a module name that matches a module included in said first build but attributes associated with said module do not match said module included in said first build, determining that said each build is not a match for said first build.

12. (Original)The method of Claim 11, further including:

for each build included in said candidate list, determining a number of matches for modules included in said each build having a matching module name and attributes of a module included in said first build;

determining a valid list of one or more matching builds, each of said builds included in said valid list having a full match for modules included in said each build with modules included in said first build, each module included in said each build having a corresponding matching module included in said first build, said name and associated attributes of said each module matching said matching module included in said first build; and

determining a maybe list of one or more builds, each of said one or more builds included in said maybe list having at least one module having a module name that does not have a matching module included in said first build, said each build including at least one module having a module name and associated attributes matching another module included in said first build, wherein said each build has an associated number of matches.

13. (Original)The method of Claim 12, further including:

if said valid list is empty, for each project, adding a build from said maybe list to said valid list in which the build added has a maximum number of matches of builds associated with said each project, a project having one or more associated builds;

performing an alternative action if said valid list is empty;

if there is only one build included in said valid list, determining that said one build matches said first build; and

if there is more than one build included in said valid list, selecting one of the more than one builds included in said valid list as being the build matching said first build.

14. (Original)The method of Claim 13, further including:

selecting one or more build associated with a software project.

15. (Original)The method of Claim 13, further including:

determining a matching build in accordance with a predetermined build selected from a list of more than one build previously registered.

16. (Currently Amended) A method ~~executed in a computer system~~ for determining code volatility, the method comprising:

extracting build information for at least two builds wherein said at least two builds are produced using a compilation process;

registering said at least two builds by storing said build information corresponding to each of said at least two builds;

identifying a first and a second of said at least two builds;

~~storing information in a database regarding said first and said second builds;~~ and

determining code volatility using said build information about said first and said second builds, said code volatility being determined using one or more metrics representing an amount of code change that has occurred between said first build and said second build.

17. (Original)The method of Claim 16, further comprising:

determining a total number of functions of said first build and said second build;

determining a percentage of functions added in accordance with said total number of functions;

determining a percentage of functions removed in accordance with said total number of functions;

determining a percentage of functions modified in accordance with said total number of functions; and

determining code volatility as a sum of said percentage of functions added, said percentage of functions removed, and said percentage of functions modified.

18. (Original) The method of Claim 17, wherein said build information is stored in a database and used in determining said code volatility.

19. (Original)The method of Claim 18, further including:

determining function checksum information used in determining differences in a function in which a first version of a function is associated with one module and a second version of the function is associated with a second module, said first module being associated with a first build, and said second module being associated with a second build.

20. (Currently Amended) A method ~~executed in a computer system~~ for tracking build information comprising:

extracting build information from one or more builds wherein said one or more builds are produced using a compilation process;

registering said one or more builds by storing said build information in a database;

testing software that includes one or more modules;

automatically determining a matching build for said one or more modules associated with one of said builds previously registered; and

determining testing associated with a build by performing a query using said information in said database.

21. (Original) The method of Claim 20, the method further including:

determining code volatility between two predetermined builds having corresponding build information stored in said database.

22. (Currently Amended) A computer program product for automatically tracking build information comprising:

machine executable code for extracting build information from one or more builds wherein said one or more builds are produced using a compilation process;

machine executable code for registering said one or more builds by storing said build information corresponding to each of said one or more builds;

machine executable code for determining ~~at~~ runtime software module information about a version of software being tested; and

machine executable code for determining a first of said one or more builds corresponding to said runtime software module information.

23. (Original) The computer program product of Claim 22, further including:

machine executable code for storing build information in a database; and

machine executable code for using said build information in said database to automatically determine said first build corresponding to said runtime software module information.

24. (Original)The computer program product of Claim 23, wherein said database that includes said build information is an object database, and the computer program product further comprises:

machine executable code for creating and storing one or more objects corresponding to each of said builds; and

machine executable code for creating and storing one or more objects corresponding to software modules included in each of the builds.

25. (Original)The computer program product of Claim 24, further including:

machine executable code for creating and storing a session object corresponding to a test session of said version of software;

machine executable code for creating and storing one or more objects corresponding to software modules describing said runtime software module information;

machine executable code for determining automatically a previously created build object corresponding to one of said one or more builds previously registered; and

machine executable code for storing an address of said previously created build object in said session object.

26. (Original)The computer program product of Claim 22, wherein said machine executable code for registering one or more builds includes:

machine executable code for storing software module information about at least one module for each of said one or more builds.

27. (Original)The computer program product of Claim 22, wherein said machine executable code for automatically determining at runtime software module information about a version of software being tested further includes:

machine executable code for gathering runtime information about software modules dynamically loaded in said computer system.

28. (Original)The computer program product of Claim 27, wherein machine executable code associated with an operating system is used in gathering runtime information.

29. (Original) The computer program product of Claim 23, further including:

machine executable code for using said build information in said database to automatically determine a second of said one or more builds corresponding to software module information included in a bug report; and

machine executable code for creating and storing a bug report object corresponding to said bug report, said bug report object including an address associated with said second build.

30. (Original)The computer program product of Claim 29, further including:

machine executable code for submitting a bug report included in a formatted electronic message; and

machine executable code for interpreting data included in said formatted electronic message to enable determination of said second build.

31. (Original)The computer program product of Claim 30, further including:

machine executable code for creating and storing another build object corresponding to a build in which said bug report is identified as being corrected; and

machine executable code for associating said other build object with said bug report object in said database.

32. (Original)The computer program product of Claim 22, wherein said machine executable code for automatically determining said first build further comprises:

machine executable code for determining said first build for which a matching build is being determined from said one or more builds registered;

machine executable code for determining a candidate list including one or more builds having at least one module in common with said first build;

machine executable code for determining, for each build included in said candidate list, if modules included in said each build match modules included in said first build;

machine executable code for determining, for each build included in said candidate list, if there are no modules included in said each build that have a module name and associated attributes matching a module included in said first build, determining that said each build is not a match for said first build; and

machine executable code for determining, for each build included in said candidate list, that said each build is not a match for said first build if a first of said modules included in said each build has a module name that matches a module included in said first build but attributes associated with said module do not match said module included in said first build.

33. (Original)The computer program product of Claim 22, further including:

machine executable code for determining, for each build included in said candidate list, a number of matches for modules included in said each build having a matching module name and attributes of a module included in said first build;

machine executable code for determining a valid list of one or more matching builds, each of said builds included in said valid list having a full match for modules included in said each build with modules included in said first build, each module included in said each build having a corresponding matching module included in said first build, said name and associated attributes of said each module matching said matching module included in said first build; and

machine executable code for determining a maybe list of one or more builds, each of said one or more builds included in said maybe list having at least one module having a module name that does not have a matching module included in said first build, said each build including at least one module having a module name and associated attributes matching another module included in said first build, wherein said each build has an associated number of matches.

34. (Original)The computer program product of Claim 33, further including:

machine executable code for adding, if said valid list is empty, for each project, a build from said maybe list to said valid list in which the build added has a maximum number of matches of builds associated with said each project, a project having one or more associated builds;

machine executable code for performing an alternative action if said valid list is empty;

machine executable code for determining that said one build matches said first build if there is only one build included in said valid list; and

machine executable code for selecting one of the more than one builds included in said valid list as being the build matching said first build if there is more than one build included in said valid list.

35. (Original)The computer program product of Claim 34, further including:

machine executable code for selecting one or more build associated with a software project.

36. (Original)The computer program product of Claim 34, further including:

machine executable code for determining a matching build in accordance with a predetermined build selected from a list of more than one build previously registered.

37. (Currently Amended) A computer program product for determining code volatility comprising:

machine executable code for extracting build information from at least two builds wherein said at least two builds are produced using a compilation process;

machine executable code for registering said at least two builds by storing said build information corresponding to each of said at least two builds;

machine executable code for identifying a first and a second of said at least two builds;

~~machine executable code for storing information in a database regarding said first and said second builds;~~ and

machine executable code for determining code volatility using said build information about said first and said second builds, said code volatility being determined using one or more metrics representing an amount of code change that has occurred between said first build and said second build.

38. (Original)The computer program product of Claim 37, further comprising:

machine executable code for determining a total number of functions of said first build and said second build;

machine executable code for determining a percentage of functions added in accordance with said total number of functions;

machine executable code for determining a percentage of functions removed in accordance with said total number of functions;

machine executable code for determining a percentage of functions modified in accordance with said total number of functions; and

machine executable code for determining code volatility as a sum of said percentage of functions added, said percentage of functions removed, and said percentage of functions modified.

39. (Original) The computer program product of Claim 38, wherein said build information is stored in a database and used in determining said code volatility.

40. (Original)The computer program product of Claim 39, further including:

machine executable code for determining function signature information used in determining differences in a function in which a first version of a function is associated with one module and a second version of the function is associated with a second module, said first module being associated with a first build, and said second module being associated with a second build.

41. (Currently Amended) A computer program product for tracking build information comprising:

machine executable code for extracting build information from one or more builds wherein said one or more builds are produced using a compilation process;

machine executable code for registering said one or more builds by storing said build information in a database;

machine executable code for testing software that includes one or more modules;

machine executable code for automatically determining a matching build for said one or more modules associated with one of said builds previously registered; and

machine executable code for determining testing associated with a build by performing a query using said information in said database.

42. (Original) The computer program product of Claim 41, the computer program product further including:

machine executable code for determining code volatility between two predetermined builds having corresponding build information stored in said database.